



\*\*FILE\*\*ID\*\*BOOSTRING

F 12

800  
V04

BBBBBBBB	000000	000000	SSSSSSSS	TTTTTTTT	RRRRRRRR	IIIIII	NN	NN	GGGGGGGG
BBBBBBBB	000000	000000	SSSSSSSS	TTTTTTTT	RRRRRRRR	IIIIII	NN	NN	GGGGGGGG
BB	BB	00	00	00	RR	RR	NN	NN	GG
BB	BB	00	00	00	RR	RR	NN	NN	GG
BB	BB	00	00	00	RR	RR	NNNN	NNNN	GG
BB	BB	00	00	00	RR	RR	NNNN	NNNN	GG
BBBBBBBB	00	00	00	SS	TT	RRRRRRRR	NN	NN	GG
BBBBBBBB	00	00	00	SS	TT	RRRRRRRR	NN	NN	GG
BB	BB	00	00	00	SS	RR	NN	NN	GG
BB	BB	00	00	00	SS	RR	NN	NN	GG
BB	BB	00	00	00	SS	RR	NN	NN	GG
BB	BB	00	00	00	SS	RR	NN	NN	GG
BBBBBBBB	000000	000000	SSSSSSSS	TT	RR	IIIIII	NN	NN	GGGGGG
BBBBBBBB	000000	000000	SSSSSSSS	TT	RR	IIIIII	NN	NN	GGGGGG

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLL	IIIIII	SSSSSSSS

(2)	89	Miscellaneous Notes
(3)	146	DECLARATIONS
(4)	190	Conditional Assembly Parameters
(7)	683	VAX\$CMPC3 - Compare Characters (3 Operand)
(8)	787	VAX\$CMPC5 - Compare Characters (5 Operand)
(11)	1152	VAX\$LOCC - Locate Character

00000001 0000 1      BOOT\_SWITCH = 1      ; Include bootstrap emulation subset

0000 1 .NOSHOW CONDITIONALS  
0000 5 .TITLE BOO\$STRING  
0000 7 .IDENT /V04-001/ Subset Instruction Emulation for VMB and SYSBOOT  
0000 8  
0000 9 :  
0000 10 :\*\*\*\*\*  
0000 11 :\*  
0000 12 :\* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
0000 13 :\* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
0000 14 :\* ALL RIGHTS RESERVED.  
0000 15 :\*  
0000 16 :\* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
0000 17 :\* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
0000 18 :\* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
0000 19 :\* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
0000 20 :\* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
0000 21 :\* TRANSFERRED.  
0000 22 :\*  
0000 23 :\* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
0000 24 :\* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
0000 25 :\* CORPORATION.  
0000 26 :\*  
0000 27 :\* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
0000 28 :\* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
0000 29 :\*  
0000 30 :\*  
0000 31 :\*\*\*\*\*  
0000 32 :  
0000 33 :  
0000 34 :++  
0000 35 : Facility:  
0000 36 :  
0000 37 : VAX-11 Instruction Emulator  
0000 38 :  
0000 39 : Abstract:  
0000 40 :  
0000 41 : The routines in this module emulate the VAX-11 string instructions.  
0000 42 : These procedures can be a part of an emulator package or can be  
0000 43 : called directly after the input parameters have been loaded into  
0000 44 : the architectural registers.  
0000 45 :  
0000 46 : The input parameters to these routines are the registers that  
0000 47 : contain the intermediate instruction state.  
0000 48 :  
0000 49 : Environment:  
0000 50 :  
0000 51 : These routines run at any access mode, at any IPL, and are AST  
0000 52 : reentrant.  
0000 53 :  
0000 54 : Author:  
0000 55 :  
0000 56 : Lawrence J. Kenah  
0000 57 :  
0000 58 : Creation Date:  
0000 59 :  
0000 60 : 16 August 1982  
0000 61 :  
0000 62 :  
0000 63 :  
0000 64 :  
0000 65 :  
0000 66 :  
0000 67 :  
0000 68 :  
0000 69 :  
0000 70 :  
0000 71 :  
0000 72 :  
0000 73 :  
0000 74 :  
0000 75 :  
0000 76 :  
0000 77 :  
0000 78 :  
0000 79 :  
0000 80 :  
0000 81 :  
0000 82 :  
0000 83 :  
0000 84 :  
0000 85 :  
0000 86 :  
0000 87 :  
0000 88 :  
0000 89 :  
0000 90 :  
0000 91 :  
0000 92 :  
0000 93 :  
0000 94 :  
0000 95 :  
0000 96 :  
0000 97 :  
0000 98 :  
0000 99 :  
0000 100 :  
0000 101 :  
0000 102 :  
0000 103 :  
0000 104 :  
0000 105 :  
0000 106 :  
0000 107 :  
0000 108 :  
0000 109 :  
0000 110 :  
0000 111 :  
0000 112 :  
0000 113 :  
0000 114 :  
0000 115 :  
0000 116 :  
0000 117 :  
0000 118 :  
0000 119 :  
0000 120 :  
0000 121 :  
0000 122 :  
0000 123 :  
0000 124 :  
0000 125 :  
0000 126 :  
0000 127 :  
0000 128 :  
0000 129 :  
0000 130 :  
0000 131 :  
0000 132 :  
0000 133 :  
0000 134 :  
0000 135 :  
0000 136 :  
0000 137 :  
0000 138 :  
0000 139 :  
0000 140 :  
0000 141 :  
0000 142 :  
0000 143 :  
0000 144 :  
0000 145 :  
0000 146 :  
0000 147 :  
0000 148 :  
0000 149 :  
0000 150 :  
0000 151 :  
0000 152 :  
0000 153 :  
0000 154 :  
0000 155 :  
0000 156 :  
0000 157 :  
0000 158 :  
0000 159 :  
0000 160 :  
0000 161 :  
0000 162 :  
0000 163 :  
0000 164 :  
0000 165 :  
0000 166 :  
0000 167 :  
0000 168 :  
0000 169 :  
0000 170 :  
0000 171 :  
0000 172 :  
0000 173 :  
0000 174 :  
0000 175 :  
0000 176 :  
0000 177 :  
0000 178 :  
0000 179 :  
0000 180 :  
0000 181 :  
0000 182 :  
0000 183 :  
0000 184 :  
0000 185 :  
0000 186 :  
0000 187 :  
0000 188 :  
0000 189 :  
0000 190 :  
0000 191 :  
0000 192 :  
0000 193 :  
0000 194 :  
0000 195 :  
0000 196 :  
0000 197 :  
0000 198 :  
0000 199 :  
0000 200 :  
0000 201 :  
0000 202 :  
0000 203 :  
0000 204 :  
0000 205 :  
0000 206 :  
0000 207 :  
0000 208 :  
0000 209 :  
0000 210 :  
0000 211 :  
0000 212 :  
0000 213 :  
0000 214 :  
0000 215 :  
0000 216 :  
0000 217 :  
0000 218 :  
0000 219 :  
0000 220 :  
0000 221 :  
0000 222 :  
0000 223 :  
0000 224 :  
0000 225 :  
0000 226 :  
0000 227 :  
0000 228 :  
0000 229 :  
0000 230 :  
0000 231 :  
0000 232 :  
0000 233 :  
0000 234 :  
0000 235 :  
0000 236 :  
0000 237 :  
0000 238 :  
0000 239 :  
0000 240 :  
0000 241 :  
0000 242 :  
0000 243 :  
0000 244 :  
0000 245 :  
0000 246 :  
0000 247 :  
0000 248 :  
0000 249 :  
0000 250 :  
0000 251 :  
0000 252 :  
0000 253 :  
0000 254 :  
0000 255 :  
0000 256 :  
0000 257 :  
0000 258 :  
0000 259 :  
0000 260 :  
0000 261 :  
0000 262 :  
0000 263 :  
0000 264 :  
0000 265 :  
0000 266 :  
0000 267 :  
0000 268 :  
0000 269 :  
0000 270 :  
0000 271 :  
0000 272 :  
0000 273 :  
0000 274 :  
0000 275 :  
0000 276 :  
0000 277 :  
0000 278 :  
0000 279 :  
0000 280 :  
0000 281 :  
0000 282 :  
0000 283 :  
0000 284 :  
0000 285 :  
0000 286 :  
0000 287 :  
0000 288 :  
0000 289 :  
0000 290 :  
0000 291 :  
0000 292 :  
0000 293 :  
0000 294 :  
0000 295 :  
0000 296 :  
0000 297 :  
0000 298 :  
0000 299 :  
0000 300 :  
0000 301 :  
0000 302 :  
0000 303 :  
0000 304 :  
0000 305 :  
0000 306 :  
0000 307 :  
0000 308 :  
0000 309 :  
0000 310 :  
0000 311 :  
0000 312 :  
0000 313 :  
0000 314 :  
0000 315 :  
0000 316 :  
0000 317 :  
0000 318 :  
0000 319 :  
0000 320 :  
0000 321 :  
0000 322 :  
0000 323 :  
0000 324 :  
0000 325 :  
0000 326 :  
0000 327 :  
0000 328 :  
0000 329 :  
0000 330 :  
0000 331 :  
0000 332 :  
0000 333 :  
0000 334 :  
0000 335 :  
0000 336 :  
0000 337 :  
0000 338 :  
0000 339 :  
0000 340 :  
0000 341 :  
0000 342 :  
0000 343 :  
0000 344 :  
0000 345 :  
0000 346 :  
0000 347 :  
0000 348 :  
0000 349 :  
0000 350 :  
0000 351 :  
0000 352 :  
0000 353 :  
0000 354 :  
0000 355 :  
0000 356 :  
0000 357 :  
0000 358 :  
0000 359 :  
0000 360 :  
0000 361 :  
0000 362 :  
0000 363 :  
0000 364 :  
0000 365 :  
0000 366 :  
0000 367 :  
0000 368 :  
0000 369 :  
0000 370 :  
0000 371 :  
0000 372 :  
0000 373 :  
0000 374 :  
0000 375 :  
0000 376 :  
0000 377 :  
0000 378 :  
0000 379 :  
0000 380 :  
0000 381 :  
0000 382 :  
0000 383 :  
0000 384 :  
0000 385 :  
0000 386 :  
0000 387 :  
0000 388 :  
0000 389 :  
0000 390 :  
0000 391 :  
0000 392 :  
0000 393 :  
0000 394 :  
0000 395 :  
0000 396 :  
0000 397 :  
0000 398 :  
0000 399 :  
0000 400 :  
0000 401 :  
0000 402 :  
0000 403 :  
0000 404 :  
0000 405 :  
0000 406 :  
0000 407 :  
0000 408 :  
0000 409 :  
0000 410 :  
0000 411 :  
0000 412 :  
0000 413 :  
0000 414 :  
0000 415 :  
0000 416 :  
0000 417 :  
0000 418 :  
0000 419 :  
0000 420 :  
0000 421 :  
0000 422 :  
0000 423 :  
0000 424 :  
0000 425 :  
0000 426 :  
0000 427 :  
0000 428 :  
0000 429 :  
0000 430 :  
0000 431 :  
0000 432 :  
0000 433 :  
0000 434 :  
0000 435 :  
0000 436 :  
0000 437 :  
0000 438 :  
0000 439 :  
0000 440 :  
0000 441 :  
0000 442 :  
0000 443 :  
0000 444 :  
0000 445 :  
0000 446 :  
0000 447 :  
0000 448 :  
0000 449 :  
0000 450 :  
0000 451 :  
0000 452 :  
0000 453 :  
0000 454 :  
0000 455 :  
0000 456 :  
0000 457 :  
0000 458 :  
0000 459 :  
0000 460 :  
0000 461 :  
0000 462 :  
0000 463 :  
0000 464 :  
0000 465 :  
0000 466 :  
0000 467 :  
0000 468 :  
0000 469 :  
0000 470 :  
0000 471 :  
0000 472 :  
0000 473 :  
0000 474 :  
0000 475 :  
0000 476 :  
0000 477 :  
0000 478 :  
0000 479 :  
0000 480 :  
0000 481 :  
0000 482 :  
0000 483 :  
0000 484 :  
0000 485 :  
0000 486 :  
0000 487 :  
0000 488 :  
0000 489 :  
0000 490 :  
0000 491 :  
0000 492 :  
0000 493 :  
0000 494 :  
0000 495 :  
0000 496 :  
0000 497 :  
0000 498 :  
0000 499 :  
0000 500 :  
0000 501 :  
0000 502 :  
0000 503 :  
0000 504 :  
0000 505 :  
0000 506 :  
0000 507 :  
0000 508 :  
0000 509 :  
0000 510 :  
0000 511 :  
0000 512 :  
0000 513 :  
0000 514 :  
0000 515 :  
0000 516 :  
0000 517 :  
0000 518 :  
0000 519 :  
0000 520 :  
0000 521 :  
0000 522 :  
0000 523 :  
0000 524 :  
0000 525 :  
0000 526 :  
0000 527 :  
0000 528 :  
0000 529 :  
0000 530 :  
0000 531 :  
0000 532 :  
0000 533 :  
0000 534 :  
0000 535 :  
0000 536 :  
0000 537 :  
0000 538 :  
0000 539 :  
0000 540 :  
0000 541 :  
0000 542 :  
0000 543 :  
0000 544 :  
0000 545 :  
0000 546 :  
0000 547 :  
0000 548 :  
0000 549 :  
0000 550 :  
0000 551 :  
0000 552 :  
0000 553 :  
0000 554 :  
0000 555 :  
0000 556 :  
0000 557 :  
0000 558 :  
0000 559 :  
0000 560 :  
0000 561 :  
0000 562 :  
0000 563 :  
0000 564 :  
0000 565 :  
0000 566 :  
0000 567 :  
0000 568 :  
0000 569 :  
0000 570 :  
0000 571 :  
0000 572 :  
0000 573 :  
0000 574 :  
0000 575 :  
0000 576 :  
0000 577 :  
0000 578 :  
0000 579 :  
0000 580 :  
0000 581 :  
0000 582 :  
0000 583 :  
0000 584 :  
0000 585 :  
0000 586 :  
0000 587 :  
0000 588 :  
0000 589 :  
0000 590 :  
0000 591 :  
0000 592 :  
0000 593 :  
0000 594 :  
0000 595 :  
0000 596 :  
0000 597 :  
0000 598 :  
0000 599 :  
0000 600 :  
0000 601 :  
0000 602 :  
0000 603 :  
0000 604 :  
0000 605 :  
0000 606 :  
0000 607 :  
0000 608 :  
0000 609 :  
0000 610 :  
0000 611 :  
0000 612 :  
0000 613 :  
000

0000 62 : Modified by:  
0000 63 :  
0000 64 : V04-001 LJK0044 Lawrence J. Kenah 6-Sep-1984  
0000 65 : The backup code for MOVTC when moving in the forward direction  
0000 66 : also needs to be changed (see LJK0039) based on the relative  
0000 67 : sizes of the source and destination strings.  
0000 68 :  
0000 69 : V01-005 KDM0107 Kathleen D. Morse 21-Aug-1984  
0000 70 : Fix bug in CMPC3. Return C clear if string length is 0.  
0000 71 :  
0000 72 : V01-004 LJK0039 Lawrence J. Kenah 20-Jul-1984  
0000 73 : Modify MOVTC backup code to reflect differences in register  
0000 74 : contents when traversing strings backwards. There are two  
0000 75 : cases based on the relative sizes of source and destination.  
0000 76 :  
0000 77 : V01-003 LJK0026 Lawrence J. Kenah 19-Mar-1984  
0000 78 : Final cleanup pass. Access violation handler is now called  
0000 79 : STRING\$ACCVIO. Set PACK\_M\$ACCVIO bit in R1 before passing  
0000 80 : control to VAX\$REFLECT\_FAULT.  
0000 81 :  
0000 82 : V01-002 LJK0011 Lawrence J. Kenah 8-Nov-1983  
0000 83 : Fix three minor bugs in MOVTC and MOVTUC. Change exception  
0000 84 : handling to reflect changed implementation.  
0000 85 :  
0000 86 : V01-001 Original Lawrence J. Kenah 16-Aug-1982  
0000 87 :--

0000 89 .SUBTITLE Miscellaneous Notes  
 0000 90 :+  
 0000 91 The following notes apply to most or all of the routines that appear in  
 0000 92 this module. The comments appear here to avoid duplication in each routine.  
 0000 93  
 0000 94 1. The VAX Architecture Standard (DEC STD 032) is the ultimate authority on  
 0000 95 the functional behavior of these routines. A summary of each instruction  
 0000 96 that is emulated appears in the Functional Description section of each  
 0000 97 routine header.  
 0000 98  
 0000 99 2. One design goal that affects the algorithms used is that these instructions  
 0000 100 can incur exceptions such as access violations that will be reported to  
 0000 101 users in such a way that the exception appears to have originated at the  
 0000 102 site of the reserved instruction rather than within the emulator. This  
 0000 103 constraint affects the algorithms available and dictates specific  
 0000 104 implementation decisions.  
 0000 105  
 0000 106 3. Each routine header contains a picture of the register usage when it is  
 0000 107 necessary to store the intermediate state of an instruction (routine) while  
 0000 108 servicing an exception.  
 0000 109  
 0000 110 The delta-PC field is used by the condition handler jacket to these  
 0000 111 routines when it determines that an exception such as an access violation  
 0000 112 occurred in response to an explicit use of one of the reserved  
 0000 113 instructions. These routines can also be called directly with the input  
 0000 114 parameters correctly placed in registers. The delta-PC field is not used in  
 0000 115 this case.  
 0000 116  
 0000 117 Note that the input parameters to any routine are a subset of the  
 0000 118 intermediate state picture.  
 0000 119  
 0000 120 Fields that are not used either as input parameters or to store  
 0000 121 intermediate state are indicated thus, XXXXX.  
 0000 122  
 0000 123 4. In the Input Parameter List for each routine, certain register fields that  
 0000 124 are not used may be explicitly listed for one reason or another. These  
 0000 125 unused input parameters are described as IRRELEVANT.  
 0000 126  
 0000 127 5. In general, the final condition code settings are determined as the side  
 0000 128 effect of one of the last instructions that executes before control is  
 0000 129 passed back to the caller with an RSB. It is seldom necessary to explicitly  
 0000 130 manipulate condition codes with a BIxPSW instruction or similar means.  
 0000 131  
 0000 132 6. There is only a small set of exceptions that are reflected to the user in an  
 0000 133 altered fashion, with the exception PC changed from within the emulator to  
 0000 134 the site of the original entry into these routines. The instructions that  
 0000 135 generate these exceptions are all immediately preceded by a  
 0000 136  
 0000 137 MARK\_POINT yyyy\_N  
 0000 138  
 0000 139 where yyyy is the instruction name and N is a small integer. These names  
 0000 140 map directly into instruction- and context-specific routines (located at  
 0000 141 the end of this module) that put each instruction (routine) into a  
 0000 142 consistent state before passing control to a more general exception handler  
 0000 143 in a different module.  
 0000 144 :-

```
0000 146 .SUBTITLE DECLARATIONS
0000 147
0000 148 ; Include files:
0000 149
0000 150 $PSLDEF ; Define bit fields in PSL
0000 151
0000 152 .NOCROSS ; No cross reference for these
0000 153 .ENABLE SUPPRESSION ; No symbol table entries either
0000 154
0000 155 PACK_DEF ; Stack usage for exception handling
0000 156
0000 157 .DISABLE SUPPRESSION ; Turn on symbol table again
0000 158 .CROSS ; Cross reference is OK now
0000 159
0000 160 : Macro Definitions
0000 161
0000 162 .MACRO INCLUDE OPCODE BOOT_FLAG
0000 163 .IF NOT DEFINED BOOT_SWITCH
0000 164 OPCODE' DEF
0000 165 INCLUDE_`OPCODE = 0
0000 166 .IF_FALSE
0000 167 .IF IDENTICAL <BOOT_FLAG> , BOOT
0000 168 OPCODE' DEF
0000 169 INCLUDE_`OPCODE = 0
0000 170 .ENDC
0000 171 .ENDC
0000 172 .ENDM _INCLUDE
0000 173
0000 174 : External declarations
0000 175
0000 176 .DISABLE GLOBAL
0000 177
0000 181
0000 182 : PSECT Declarations:
0000 183
0000 184 .DEFAULT DISPLACEMENT , WORD
0000 185
00000000 186 .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, LONG
0000 187
0000 188 BEGIN_MARK_POINT ; Set up exception mark points
```

0000 190 .SUBTITLE Conditional Assembly Parameters  
0000 191 :+  
0000 192 Functional Description:  
0000 193 :  
0000 194 It is possible to create a subset emulator, one that emulates  
0000 195 specific reserved instructions. This capability is currently exploited  
0000 196 to create a subset emulator for use by the bootstrap programs.  
0000 197 :  
0000 198 An instruction is included in the full emulator by making an entry  
0000 199 in the following table. If the optional second parameter is present  
0000 200 and equal to BOOT, then that instruction is included in the subset  
0000 201 emulator used by the bootstrap code.  
0000 202 :-  
0000 203 :  
0000 204 .NOCROSS ; No cross reference for these  
0000 205 .ENABLE SUPPRESSION ; No symbol table entries either  
0000 206 :  
0000 207 -INCLUDE MOVTC  
0000 208 -INCLUDE MOVTUC  
0000 209 -INCLUDE CMPC3 : BOOT  
0000 210 -INCLUDE CMPC5 : BOOT  
0000 211 -INCLUDE SCANC  
0000 212 -INCLUDE SPANC  
0000 213 -INCLUDE LOCC , BOOT  
0000 214 -INCLUDE SKPC  
0000 215 -INCLUDE MATCHC  
0000 216 -INCLUDE CRC  
0000 217 :  
0000 218 .DISABLE SUPPRESSION ; Turn on symbol table again  
0000 219 .CROSS ; Cross reference is OK now  
0000 220 :  
0000 221 .NOSHOW CONDITIONALS  
0000 222 :

0000 683 .SUBTITLE VAX\$CMPC3 - Compare Characters (3 Operand)  
 0000 684 :+  
 0000 685 Functional Description:  
 0000 686  
 0000 687 The bytes of string 1 specified by the length and address 1 operands are  
 0000 688 compared with the bytes of string 2 specified by the length and address  
 0000 689 2 operands. Comparison proceeds until inequality is detected or all the  
 0000 690 bytes of the strings have been examined. Condition codes are affected  
 0000 691 by the result of the last byte comparison. Two zero length strings  
 0000 692 compare equal (i.e. Z is set and N, V, and C are cleared).  
 0000 693  
 0000 694 Input Parameters:  
 0000 695  
 0000 696 R0<15:0> = len Length of character strings  
 0000 697 R1 = src1addr Address of first character string (called S1)  
 0000 698 R3 = src2addr Address of second character string (called S2)  
 0000 699  
 0000 700 Intermediate State:  
 0000 701  
 0000 702 31 23 15 07 00  
 0000 703 +-----+-----+-----+-----+-----+  
 0000 704 delta-PC | XXXX | Len : R0  
 0000 705 +-----+-----+-----+-----+-----+  
 0000 706 src1addr : R1  
 0000 707 +-----+-----+-----+-----+-----+  
 0000 708 XXXXX : R2  
 0000 709 +-----+-----+-----+-----+-----+  
 0000 710 src2addr : R3  
 0000 711 +-----+-----+-----+-----+-----+  
 0000 712  
 0000 713 Output Parameters:  
 0000 714  
 0000 715 Strings are IDENTICAL  
 0000 716  
 0000 717 R0 = 0  
 0000 718 R1 = Address of one byte beyond end of S1  
 0000 719 R2 = 0 (same as R0)  
 0000 720 R1 = Address of one byte beyond end of S2  
 0000 721  
 0000 722 Strings DO NOT MATCH  
 0000 723  
 0000 724 R0 = Number of bytes left in strings (including first byte  
 0000 725 that did not match)  
 0000 726 R1 = Address of nonmatching byte in S1  
 0000 727 R2 = R0  
 0000 728 R3 = Address of nonmatching byte in S2  
 0000 729  
 0000 730 Condition Codes:  
 0000 731  
 0000 732 In general, the condition codes reflect whether or not the strings  
 0000 733 are considered the same or different. In the case of different  
 0000 734 strings, the condition codes reflect the result of the comparison  
 0000 735 that indicated that the strings are not equal.  
 0000 736  
 0000 737 Strings are IDENTICAL  
 0000 738  
 0000 739 N <- 0

```

0000 740 :          Z <- 1          ; (byte in S1) EQL (byte in S2)
0000 741 :          V <- 0
0000 742 :          C <- 0
0000 743 :          Strings DO NOT MATCH
0000 744 :          N <- (byte in S1) LSS (byte in S2)
0000 745 :          Z <- 0          ; (byte in S1) NEQ (byte in S2)
0000 746 :          V <- 0
0000 747 :          C <- (byte in S1) LSSU (byte in S2)
0000 748 :          where "byte in S1" or "byte in S2" may indicate the fill character
0000 749 :          Side Effects:
0000 750 :          This routine uses one longword of stack.
0000 751 :          752 :-
0000 752 :          753 : Side Effects:
0000 753 :          754 : This routine uses one longword of stack.
0000 754 :          755 :-
0000 755 :          756 :-
0000 756 :          757 :          758 VAX$CMPC3:-
50 50 3C 0000 759 MOVZWL R0,R0          ; Clear unused bits & check for zero
0D 13 0003 760 BEQL 20$                ; Simply return if zero length string
0005 761
5A DD 0005 762 PUSHL R10              ; Save R10 so it can hold handler
0007 763 ESTABLISH_HANDLER -          ; Store address of condition handler
0007 764 STRING_ACCVIO -            ; Store address of condition handler
0007 765
81 83 91 0007 766 MARK_POINT CMPC3_1
0B 12 000A 767 10$: CMPB (R3)+,(R1)+ ; Character match?
F8 50 F5 000C 768 BNEQ 30$            ; Exit loop if different
000F 769 SOBGTR R0,10$              ; Exit path for strings IDENTICAL (R0 = 0, either on input or after loop)
000F 770
000F 771 : Exit path for strings IDENTICAL (R0 = 0, either on input or after loop)
000F 772
5A 8E D0 000F 773 MOVL (SP)+,R10          ; Restore saved R10
52 D4 0012 774 20$: CLRL R2              ; Set R2 for output value of 0
50 D5 0014 775 TSTL R0              ; Set condition codes
05 0016 776 RSB               ; Return point for IDENTICAL strings
0017 777
0017 778 : Exit path when strings DO NOT MATCH
0017 779
5A 8E D0 0017 780 30$: MOVL (SP)+,R10          ; Restore saved R10
52 50 D0 001A 781 MOVL R0,R2              ; R0 and R2 are the same on exit
73 71 91 001D 782 CMPB -(R1),-(R3)          ; Reset R1 and R3 and set condition codes
05 0020 783 RSB               ; Return point when strings DO NOT MATCH

```

0021 787 .SUBTITLE VAX\$CMPC5 - Compare Characters (5 Operand)  
 0021 788 :+  
 0021 789 Functional Description:  
 0021 790  
 0021 791 The bytes of the string 1 specified by the length 1 and address 1  
 0021 792 operands are compared with the bytes of the string 2 specified by the  
 0021 793 length 2 and address 2 operands. If one string is longer than the  
 0021 794 other, the shorter string is conceptually extended to the length of the  
 0021 795 longer by appending (at higher addresses) bytes equal to the fill  
 0021 796 operand. Comparison proceeds until inequality is detected or all the  
 0021 797 bytes of the strings have been examined. Condition codes are affected  
 0021 798 by the result of the last byte comparison. Two zero length strings  
 0021 799 compare equal (i.e. Z is set and N, V, and C are cleared).  
 0021 800  
 0021 801 Input Parameters:  
 0021 802  
 0021 803 R0<15:0> = len Length of first character string (called S1)  
 0021 804 R0<23:16> = fill Fill character that is used when strings have  
 0021 805 different lengths  
 0021 806 R1 = addr Address of first character string  
 0021 807 R2<15:0> = len Length of second character string (called S2)  
 0021 808 R3 = addr Address of second character string  
 0021 809  
 0021 810 Intermediate State:  
 0021 811  
 0021 812 31 23 15 07 00  
 0021 813 +-----+-----+-----+-----+-----+  
 0021 814 | delta-PC | fill | src1len | : R0  
 0021 815 +-----+-----+-----+-----+-----+  
 0021 816 | src1addr | : R1  
 0021 817 +-----+-----+-----+-----+-----+  
 0021 818 | XXXXX | src2len | : R2  
 0021 819 +-----+-----+-----+-----+-----+  
 0021 820 | src2addr | : R3  
 0021 821 +-----+-----+-----+-----+-----+  
 0021 822  
 0021 823 Output Parameters:  
 0021 824  
 0021 825 Strings are IDENTICAL  
 0021 826  
 0021 827 R0 = 0  
 0021 828 R1 = Address of one byte beyond end of S1  
 0021 829 R2 = 0 (same as R0)  
 0021 830 R3 = Address of one byte beyond end of S2  
 0021 831  
 0021 832 Strings DO NOT MATCH  
 0021 833  
 0021 834 R0 = Number of bytes remaining in S1 when mismatch detected  
 0021 835 (or zero if S1 exhausted before mismatch detected)  
 0021 836 R1 = Address of nonmatching byte in S1  
 0021 837 R2 = Number of bytes remaining in S2 when mismatch detected  
 0021 838 (or zero if S2 exhausted before mismatch detected)  
 0021 839 R3 = Address of nonmatching byte in S2  
 0021 840  
 0021 841 Condition Codes:  
 0021 842  
 0021 843 In general, the condition codes reflect whether or not the strings

0021 844 ; are considered the same or different. In the case of different  
0021 845 ; strings, the condition codes reflect the result of the comparison  
0021 846 ; that indicated that the strings are not equal.  
0021 847 ;  
0021 848 ; Strings are IDENTICAL  
0021 849 ;  
0021 850 ; N <- 0  
0021 851 ; Z <- 1 ; (byte in S1) EQL (byte in S2)  
0021 852 ; V <- 0  
0021 853 ; C <- 0  
0021 854 ;  
0021 855 ; Strings DO NOT MATCH  
0021 856 ;  
0021 857 ; N <- (byte in S1) LSS (byte in S2)  
0021 858 ; Z <- 0 ; (byte in S1) NEQ (byte in S2)  
0021 859 ; V <- 0  
0021 860 ; C <- (byte in S1) LSSU (byte in S2)  
0021 861 ;  
0021 862 ; where "byte in S1" or "byte in S2" may indicate the fill character  
0021 863 ;  
0021 864 ; Side Effects:  
0021 865 ;  
0021 866 ; This routine uses two longwords of stack.  
0021 867 ;  
0021 868 ;  
0021 869 .ENABLE LOCAL\_BLOCK  
0021 870 ;  
0021 871 VAX\$CMPC5::  
5A DD 0021 872 PUSHL R10 ; Save R10 so it can hold handler  
50 F0 54 DD 0023 873 ESTABLISH\_HANDLER -  
50 50 3C 002A 0025 874 STRING\_ACCVIO ; Store address of condition handler  
28 13 002D 875 PUSHL R4 ; Save register  
52 52 3C 002F 876 ASHL #16, R0, R4 ; Get escape character  
14 13 0032 877 MOVZWL R0, R0 ; Clear unused bits & is S1 length zero?  
0034 878 BEQL 50\$ ; Branch if yes  
0034 879 MOVZWL R2, R2 ; Clear unused bits & is S2 length zero?  
0034 880 BEQL 30\$ ;  
0034 881 ; Main loop. The following loop executes when both strings have characters  
0034 882 ; remaining and inequality has not yet been detected.  
0034 883 ;  
0034 884 ; THE FOLLOWING LOOP IS A TARGET FOR FURTHER OPTIMIZATION IN THAT THE  
0034 885 ; LOOP SHOULD NOT REQUIRE TWO SOBGTR INSTRUCTIONS. NOTE, THOUGH, THAT  
0034 886 ; THE CURRENT UNOPTIMIZED LOOP IS EASIER TO BACK UP.  
0034 887 ;  
0034 888 ;  
83 81 91 0034 889 MARK\_POINT CMPC5\_1  
09 50 F5 0037 890 10\$: CMPB (R1)+, (R3)+ ; Characters match?  
32 12 0039 891 BNEQ 80\$ ; Exit loop if bytes different  
003C 892 SOBGTR R0, 20\$ ; Check for S1 exhausted  
003C 893 ;  
003C 894 ; The next test determines whether S2 is also exhausted.  
003C 895 ;  
52 D7 003C 896 DECL R2 ; Put R2 in step with R0  
1C 12 003E 897 BNEQ 60\$ ; Branch if bytes remaining in S2  
0040 898 ;  
0040 899 ; This is the exit path for identical strings. If we get here, then both  
0040 900 ; R0 and R2 are zero. The condition codes are correctly set (by the ASHL

0040 901 ; instruction) so the registers are restored with a POPR to avoid changing  
 0040 902 ; the condition codes.  
 0040 903  
 0040 904 IDENTICAL:  
 0410 8F BA 0040 905 POPR #^M<R4,R10> ; Restore saved registers  
 05 0044 906 RSB ; Exit indicating IDENTICAL strings  
 EC 52 F5 0045 907 20\$: SOBGTR R2,10\$ ; Check for S2 exhausted  
 0048 908 : The following loop is entered when all of S2 has been processed but  
 0048 909 : there are characters remaining in S1. In other words,  
 0048 910 : R0 GTRU 0  
 0048 911 : R2 EQL 0  
 0048 912 :  
 0048 913 :  
 0048 914 :  
 0048 915 :  
 0048 916 : The remaining characters in S1 are compared to the fill character.  
 0048 917  
 54 81 91 0048 918 MARK\_POINT CMPC5\_2  
 05 12 0048 919 30\$: CMPB (R1)+,R4 ; Characters match?  
 F8 50 F5 004D 920 BNEQ 40\$ ; Exit loop if no match  
 0050 921 SOBGTR R0,30\$ ; Any more bytes in S1?  
 EE 11 0050 922 BRB IDENTICAL ; Exit indicating IDENTICAL strings  
 54 71 91 0052 923 40\$: CMPB -(R1),R4 ; Reset R1 and set condition codes  
 17 11 0055 924 BRB NO\_MATCH ; Exit indicating strings DO NOT MATCH  
 0057 925 : The following code executes if S1 has zero length on input. If S2 also  
 0057 : has zero length, the routine simply returns, indicating equal strings.  
 0057 926 :  
 0057 927 :  
 0057 928 : The following loop is entered when all of S1 has been processed but  
 0057 : there are characters remaining in S2. In other words,  
 0057 929 :  
 52 52 3C 0057 930 R0 EQL 0 ; Clear unused bits. Is S2 len also zero?  
 E4 13 005A 931 50\$: MOVZWL R2,R2 ; Exit indicating IDENTICAL strings  
 005C 932 BEQL IDENTICAL ;  
 005C 933 :  
 005C 934 : The following loop is entered when all of S1 has been processed but  
 005C : there are characters remaining in S2. In other words,  
 005C 935 :  
 005C 936 : R0 EQL 0  
 005C 937 : R2 GTRU 0  
 005C 938 :  
 005C 939 :  
 005C 940 : The remaining characters in S2 are compared to the fill character.  
 005C 941  
 83 54 91 005C 942 MARK\_POINT CMPC5\_3  
 05 12 005F 943 60\$: CMPB R4,(R3)+ ; Characters match?  
 F8 52 F5 0061 944 BNEQ 70\$ ; Exit loop if no match  
 0064 945 SOBGTR R2,60\$ ; Any more bytes in S2?  
 DA 11 0064 946 BRB IDENTICAL ; Exit indicating IDENTICAL strings  
 0066 947 :  
 73 54 91 0066 948 70\$: CMPB R4,-(R3) ; Reset R3 and set condition codes  
 03 11 0069 949 BRB NO\_MATCH ; Exit indicating strings DO NOT MATCH  
 006B 950 :  
 006B 951 : The following exit path is taken if both strings have characters  
 006B : remaining and a character pair that did not match was detected.  
 006B 952 :  
 006B 953 :  
 006B 954 :  
 73 71 91 006B 955 80\$: CMPB -(R1),-(R3) ; Reset R1 and R3 and set condition codes  
 006E 956 NO\_MATCH: POPR #^M<R4,R10> ; Restore R4 and R10  
 0410 8F BA 006E 957 : without changing condition codes

05 0072 958 RSB ; Exit indicating strings DO NOT MATCH  
0073 959  
0073 960 .DISABLE LOCAL\_BLOCK

0073 1152 .SUBTITLE VAX\$LOCC - Locate Character  
 0073 1153 :+  
 0073 1154 Functional Description:  
 0073 1155  
 0073 1156 The character operand is compared with the bytes of the string specified  
 0073 1157 by the length and address operands. Comparison continues until equality  
 0073 1158 is detected or all bytes of the string have been compared. If equality  
 0073 1159 is detected; the condition code Z-bit is cleared; otherwise the Z-bit  
 0073 1160 is set.  
 0073 1161  
 0073 1162 Input Parameters:  
 0073 1163  
 0073 1164 R0<15:0> = len Length of character string  
 0073 1165 R0<23:16> = char Character to be located  
 0073 1166 R1 = addr Address of character string  
 0073 1167  
 0073 1168 Intermediate State:  
 0073 1169  
 0073 1170 31 23 15 07 00  
 0073 1171 +-----+-----+-----+-----+-----+  
 0073 1172 | delta-PC | char | len | : R0  
 0073 1173 +-----+-----+-----+-----+-----+  
 0073 1174 | | | addr | : R1  
 0073 1175 +-----+-----+-----+-----+-----+  
 0073 1176  
 0073 1177 Output Parameters:  
 0073 1178  
 0073 1179 Character Found  
 0073 1180  
 0073 1181 R0 = Number of bytes remaining in the string (including located one)  
 0073 1182 R1 = Address of the located byte  
 0073 1183  
 0073 1184 Character NOT Found  
 0073 1185  
 0073 1186 R0 = 0  
 0073 1187 R1 = Address of one byte beyond end of string  
 0073 1188  
 0073 1189 Condition Codes:  
 0073 1190  
 0073 1191 N <- 0  
 0073 1192 Z <- R0 EQL 0  
 0073 1193 V <- 0  
 0073 1194 C <- 0  
 0073 1195  
 0073 1196 The Z bit is clear if the character is located.  
 0073 1197 The Z bit is set if the character is NOT located.  
 0073 1198  
 0073 1199 Side Effects:  
 0073 1200  
 0073 1201 This routine uses two longwords of stack.  
 0073 1202 :-  
 0073 1203  
 0073 1204 VAX\$LOCC::  
 SA DD 0073 1205 PUSHL R10 ; Save R10 so it can hold handler  
 0075 1206 ESTABLISH\_HANDLER -  
 52 DD 0075 1207 STRING\_ACCVIO ; Store address of condition handler  
 0075 1208 PUSHL R2 ; Save register

H 13

52	50	F0	8F	78	0077	1209	ASHL	#-16	R0,R2	; Get character to be located	
	50	50	3C	007C	1210		MOVZWL	R0	R0	; Clear unused bits & check for 0 length	
	08	13	007F	1211			BEQL	20\$		; Simply return if length is 0	
			0081	1212							
81	52	91	0081	1214	10\$:	MARK_POINT			LOCC_1	; Character match?	
	0A	13	0084	1215		CMPB	R2	(R1)+		; Exit loop if yes	
F8	50	F5	0086	1216		BEQL	30\$				
			0089	1217		SOBGTR	R0	10\$			
			0089	1218							
			0089	1219							
			0089	1220							
0404	8F	BA	0089	1221	20\$:	POPR	#^M<R2,R10>			; Restore saved R2 and R10	
	50	D5	008D	1222		TSTL	R0			; Insure that C-bit is clear	
		05	008F	1223		RSB				; Return with Z-bit set	
			0090	1224							
			0090	1225							
			0090	1226							
51	D7	0090	1227	30\$:	DECL	R1				; Point R1 to located character	
F5	11	0092	1228		BRB	20\$				; Join common code	

0094 2168 END\_MARK\_POINT  
0094 2169  
0094 2170 .END

BOOT\$STRING	= 00000001	OPS\$-CVTLG	= 00004EFD
IDENTICAL	= 00000040 R 02	OPS\$-CVTLH	= 00006EFD
NO MATCH	= 0000006E R 02	OPS\$-CVTLP	= 000000F9
OPS\$-ACBD	= 0000006F	OPS\$-CVTPL	= 00000036
OPS\$-ACBF	= 0000004F	OPS\$-CVTPS	= 00000008
OPS\$-ACBG	= 00004FFD	OPS\$-CVTPT	= 00000024
OPS\$-ACBH	= 00006FFD	OPS\$-CVTRDL	= 0000006B
OPS\$-ADDD2	= 00000060	OPS\$-CVTRFL	= 0000004B
OPS\$-ADDD3	= 00000061	OPS\$-CVTRGL	= 00004BFD
OPS\$-ADDF2	= 00000040	OPS\$-CVTRHL	= 00006BFD
OPS\$-ADDF3	= 00000041	OPS\$-CVTSP	= 00000009
OPS\$-ADDG2	= 000040FD	OPS\$-CVTTP	= 00000026
OPS\$-ADDG3	= 000041FD	OPS\$-CVTWD	= 0000006D
OPS\$-ADDH2	= 000060FD	OPS\$-CVTWF	= 0000004D
OPS\$-ADDH3	= 000061FD	OPS\$-CVTWG	= 00004DFD
OPS\$-ADDP4	= 00000020	OPS\$-CVTWH	= 00006DFD
OPS\$-ADDP6	= 00000021	OPS\$-DIVD2	= 00000066
OPS\$-ASHP	= 000000F8	OPS\$-DIVD3	= 00000067
OPS\$-CLRD	= 0000007C	OPS\$-DIVF2	= 00000046
OPS\$-CLRF	= 000000D4	OPS\$-DIVF3	= 00000047
OPS\$-CLRG	= 0000007C	OPS\$-DIVG2	= 000046FD
OPS\$-CLRH	= 00007CFD	OPS\$-DIVG3	= 000047FD
OPS\$-CMPD	= 00000071	OPS\$-DIVH2	= 000066FD
OPS\$-CMPP	= 00000051	OPS\$-DIVH3	= 000067FD
OPS\$-CMPG	= 000051FD	OPS\$-DIVP	= 00000027
OPS\$-CMPH	= 000071FD	OPS\$-EDITPC	= 00000038
OPS\$-CMPP3	= 00000035	OPS\$-EMODD	= 00000074
OPS\$-CMPP4	= 00000037	OPS\$-EMODF	= 00000054
OPS\$-CRC	= 0000000B	OPS\$-EMODG	= 000054FD
OPS\$-CVTBD	= 0000006C	OPS\$-EMODH	= 000074FD
OPS\$-CVTBF	= 0000004C	OPS\$-MATCHC	= 00000039
OPS\$-CVTBG	= 00004CFD	OPS\$-MNEGD	= 00000072
OPS\$-CVTBH	= 00006CFD	OPS\$-MNEGF	= 00000052
OPS\$-CVTDB	= 00000068	OPS\$-MNEGG	= 000052FD
OPS\$-CVTDF	= 00000076	OPS\$-MNEGH	= 000072FD
OPS\$-CVTDH	= 000032FD	OPS\$-MOVD	= 00000070
OPS\$-CVTDL	= 0000006A	OPS\$-MOVF	= 00000050
OPS\$-CVTDW	= 00000069	OPS\$-MOVG	= 000050FD
OPS\$-CVTFB	= 00000048	OPS\$-MOVH	= 000070FD
OPS\$-CVTFD	= 00000056	OPS\$-MOVP	= 00000034
OPS\$-CVTFG	= 000099FD	OPS\$-MOVTC	= 0000002E
OPS\$-CVTFH	= 000098FD	OPS\$-MOVTC	= 0000002F
OPS\$-CVTFL	= 0000004A	OPS\$-MULD2	= 00000064
OPS\$-CVTFW	= 00000049	OPS\$-MULD3	= 00000065
OPS\$-CVTGB	= 000048FD	OPS\$-MULF2	= 00000044
OPS\$-CVTGF	= 000033FD	OPS\$-MULF3	= 00000045
OPS\$-CVTGH	= 000056FD	OPS\$-MULG2	= 000044FD
OPS\$-CVTGL	= 00004AFD	OPS\$-MULG3	= 000045FD
OPS\$-CVTGW	= 000049FD	OPS\$-MULH2	= 000064FD
OPS\$-CVTHB	= 000068FD	OPS\$-MULH3	= 000065FD
OPS\$-CVTHD	= 0000F7FD	OPS\$-MULP	= 00000025
OPS\$-CVTHF	= 0000F6FD	OPS\$-POLYD	= 00000075
OPS\$-CVTHG	= 000076FD	OPS\$-POLYF	= 00000055
OPS\$-CVTHL	= 00006AFD	OPS\$-POLYG	= 000055FD
OPS\$-CVTHW	= 000069FD	OPS\$-POLYH	= 000075FD
OPS\$-CVTLD	= 0000006E	OPS\$-SCANC	= 0000002A
OPS\$-CVTLF	= 0000004E	OPS\$-SKPC	= 0000003B

OP\$-SPANC	= 0000002B
OP\$-SUBD2	= 00000062
OP\$-SUBD3	= 00000063
OP\$-SUBF2	= 00000042
OP\$-SUBF3	= 00000043
OP\$-SUBG2	= 000042FD
OP\$-SUBG3	= 000043FD
OP\$-SUBH2	= 000062FD
OP\$-SUBH3	= 000063FD
OP\$-SUBP4	= 00000022
OP\$-SUBP6	= 00000023
OP\$-TSTD	= 00000073
OP\$-TSTF	= 00000053
OP\$-TSTG	= 000053FD
OP\$-TSTH	= 000073FD
VAX\$CMPC3	00000000 RG 02
VAX\$CMPC5	00000021 RG 02
VAX\$LOCC	00000073 RG 02

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
ABS .	00000000	( 0.) 00 ( 0.)	NOPIC USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
\$ABSS	00000000	( 0.) 01 ( 1.)	NOPIC USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
-VAX\$CODE	00000094	( 148.) 02 ( 2.)	PIC USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	15	00:00:00.06	00:00:01.22
Command processing	74	00:00:00.73	00:00:05.99
Pass 1	390	00:00:11.56	00:00:41.58
Symbol table sort	0	00:00:00.58	00:00:01.86
Pass 2	102	00:00:05.40	00:00:15.24
Symbol table output	16	00:00:00.11	00:00:00.40
Psect synopsis output	2	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	599	00:00:18.45	00:01:06.31

The working set limit was 1500 pages.

70465 bytes (138 pages) of virtual memory were used to buffer the intermediate code.

There were 30 pages of symbol table space allocated to hold 447 non-local and 14 local symbols.

4923 source lines were read in Pass 1, producing 13 object records in Pass 2.

145 pages of virtual memory were used to define 143 macros.

-----  
! Macro library statistics !  
-----

Macro library name

-----  
-\$255\$DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1  
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2  
TOTALS (all libraries)

Macros defined

-----  
8  
5  
13

584 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:BOO\$STRING/OBJ=OBJ\$:BOO\$STRING MSRC\$:\$BOOTSWT/UPDATE=(ENH\$:\$BOOTSWT)+MSRC\$:\$MISSING/UPDATE=(ENH\$:\$MISSING)+MSRC\$:\$VAXSTRING/

0142 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

